USAISEC
US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

System Re-engineering Project Executive Summary

(ASQB-GI-92-003)

NOVEMBER 1991

AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704--188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION / AVAILABILITY OF REPORT N/A
2b. DECLASSIFICATION / DOUWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQB-GI-92-003	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION AIRMICS 6b. OFFICE SYMBOL (if applicable) ASQB-GI	7a. NAME OF MONITORING ORGANIZATION N/A
6c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	7b. ADDRESS (City, State, and Zip Code) N/A
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS 8b. OFFICE SYMBOL (if applicable) ASQB-GI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg., Georgia Institute of Technology Atlanta, GA 30332-0800	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62783A PROJECT NO.: DY10 TASK NO.: 02-04 WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)
System Re-engineering Project Executive Summary (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Reginald L. Hobbs; John Mitchell; Glenn Racine

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM _____ TO _____	1991 November 25	8

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Computer-Aided Software Engineering (CASE); Distributed Software;Ada; COBOL;Systems Analysis, Systems Design, Life Cycle Development;Functional Decomposition; Object-Oriented Design

19. ABSTRACT (Continue on reverse if necessary and identity by block number)

The purpose of this paper is to describe the system re-engineering project conducted by the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS) and the Information Systems Software Center, Software Development Center - Atlanta (SDC-A). The project was concerned with the analysis and re-engineering of a Standard Army Management Information System (STAMIS) application. This project involved reverse engineering, evaluation of structured design and object-oriented design, and re-implementation of the system in Ada. This executive summary presents the approach to re-engineering the system, the lessons learned while going through the process, and issues to be considered in future tasks of this nature.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED / UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
Reginald L. Hobbs	(404) 894-3110	ASQB-GI

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

This research was performed as an in-house project at the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS) in conjunction with personnel from the Software Development Center-Atlanta (SDC-A). The project was concerned with the analysis and re-engineering of a Standard Army Management Information System (STAMIS) application. The objectives of the project were to assess the use of CASE tools, evaluate object-oriented design versus functional decomposition, re-implement a system in Ada, determine the training requirements for COBOL software engineers, and compare the old and new systems. This executive summary is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

For more detailed information on the project, refer to *Ada Transition Research Project (Phase I)* (ASQB-GI-91-005) and *Ada Transition Research Project (Phase II)* (ASQB-GI-92-004).

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ Glenn E. Racine
Glenn E. Racine
Chief
CISD

s/ John R. Mitchell
John R. Mitchell
Director
AIRMICS

SYSTEM RE-ENGINEERING PROJECT

U. S. Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS)/Software Development Center-Atlanta(SDC-A)

1-1. ABSTRACT

The purpose of this paper is to describe the system re-engineering project conducted by the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS) and the Information Systems Software Center, Software Development Center - Atlanta (SDC-A). The project was concerned with the analysis and re-engineering of a Standard Army Management Information System (STAMIS) application. This project involved reverse engineering, evaluation of structured design and object-oriented design, and re-implementation of the system in Ada. This paper will present our approach to re-engineering the system, the lessons learned while going through the process, and issues to be considered in future tasks of this nature.

1-2. BACKGROUND

The US Army Information Systems Engineering Command (ISEC) maintains over one hundred STAMIS applications in the major functional areas of logistics, personnel, finance, and communications. Most of these systems are written in COBOL, are batch oriented, and have been operational for more than twenty years. Maintenance of these systems is costly and time consuming primarily due to the deterioration of design structure associated with many years of maintenance activity.

AIRMICS and the Software Development Center-Atlanta (SDC-A) began this project to examine techniques for modernizing old COBOL applications to reduce the maintenance effort required and to incorporate sound software engineering principles. We refer to this effort as system re-engineering since it encompassed reverse engineering to document the functional requirements, redesign to improve the structure of the system, and re-implementation to code the new system in Ada.

1-3. OBJECTIVES

The objectives of the project were to assess the use of CASE tools, evaluate object-oriented design versus functional decomposition, re-implement the system in Ada, determine the training requirements for COBOL software engineers, and compare the old and new systems.

1-4. APPROACH

The following sections describe the re-engineering approach we used as a 7-step process. The 7 steps are: 1) Application Selection, 2) Re-engineering Tool Support, 3) Reverse Engineering, 4) System Redesign, 5) Re-implementation, 6) Test and Evaluation, and 7) Installation.

1-4.1. APPLICATION SELECTION

AIRMICS, in conjunction with SDC-A, selected the Installation Materiel Condition Status Reporting System (IMCSRS) to re-engineer and redesign for this effort. IMCSRS is an operational STAMIS, written in COBOL, consisting of 10,000 lines of code in 15 programs. It had been maintained as a STAMIS application for 20 years and was being used by over 40 Army installations. We chose this system because it was representative of the typical STAMIS application (i.e. batch-oriented, report generation, COBOL source code, etc.), but was much smaller than average in size.

Selecting a system that was not extremely complex, involved few interfaces, and had a reasonable number of lines of code, allowed us to complete the project using in-house personnel and provided a manageable case study for this proof-of-principle effort. One of the questions that remains concerns the repeatability and scalability of our results. We intend to re-engineer another system to determine if the methods and techniques we utilized translate to larger systems.

There is a need for a methodical, step-by-step procedure that can evaluate the potential benefits of re-engineering based on the characteristics of an application. In a future task, we plan to look into methods for determining when and under what circumstances systems should be re-engineered. These methods will develop quantifiable measures for projecting the cost benefit of re-engineering systems in terms of the impact on the business practices of an organization.

1-4.2. RE-ENGINEERING TOOL SUPPORT

The next major step involved defining the software development environment. The CASE tool selection process was accomplished over a 3-month period. AIRMICS performed a technical evaluation of commercial CASE products that support a structured design methodology, an object-oriented design methodology, and would run on our current UNIX-based SUN workstations.

The CASE tool suite called Teamwork , by Cadre Technologies, was selected for the redesign. The specific tools chosen included utilities for performing structured analysis, structured design, and the creation of Ada structure graphs. These tools were selected primarily because they provide support for both object-oriented and functional decomposition design methods, generate template Ada code, are of relatively low cost, and were consistent with our existing workstation environment. We were not evaluating the CASE tools to determine the "best" tool for the project, rather we were interested in finding a tool that met our requirements and constraints. The CADRE tools purchased included a 2-user network license, project environment software, the structured design tools, and an Ada source builder.

Having CASE tools with a consistent look and feel helped decrease the amount of training required to use the software. The graphical user interface consisted of a mouse-driven, multi-window environment containing pop-up and pull-down menus.

The amount of formal training on the hardware/software tools for the developers and programmers included: Teamwork CASE tools (10 days) and Ada programming language (8 weeks). Informal in-house training included: Sun Workstation (2 days), UNIX Operating system (3 days), Functional Decomposition Methodology (20 days), Object-Oriented Design Methodology (20 days). Most CASE tools employ some type of system design/development methodology that the CASE vendor assumes the user understands. The formal training done through Cadre focused on the mechanics of using the CASE tool, not the underlying methodology.

Our next re-engineering task should select CASE tools that fully integrate all phases of the life cycle. Part of the difficulty in the software development was transitioning from one graphical representation to the next, while taking advantage of information already learned. Many of the transformations (particularly from the structured analysis to the structure design phase) were still largely a manual process. There was also a lack of CASE tool support to automate the reverse engineering process.

1-4.3. REVERSE ENGINEERING

Reverse engineering a system means recognizing design decisions that produced the original software and constructing a representation of these decisions to define its functional requirements. The objective of the reverse engineering phase of the project was to produce a functional description of the existing system from the COBOL source code and available documentation.

Certain stages within this analysis were diagrammed using a CASE tool (IDE's *Software Through Pictures*) for clarification of data structure and flow, but for the most part this was a manual process. The reverse engineering was completed in 3 man-months by 2 researchers and 1 administrative support clerk. The result of reverse engineering IMCSRS was a functional description document describing the existing system functionality.

One problem that became apparent during the reverse engineering process was that going into great detail analyzing the existing code had to be avoided to prevent re-implementing design errors from the original system. Another difficulty arose in that much of the initial process was done without the presence of a functional user. When the functions appeared to be complete, they were reviewed by a functional area specialist for validity. We had to redo sections of the functional description that came out of the reverse engineering study in order to accurately reflect the desired systems capabilities. As a function was reviewed and completed, it was added to the new functional description. If it was not complete, the functional area specialist further defined the actions necessary to complete the functional requirement document.

Subsequent re-engineering tasks of this type should involve the functional users at all stages of the life cycle. It is through this involvement that the functionality of the existing system and the business practices are improved. Using rapid prototyping tools, particularly to simulate the user interface for a system, will further allow for the validation of the functional description prior to redesign.

1-4.4. SYSTEM REDESIGN

The objective of the design phase of the project was to produce a detailed design of the system for implementation in Ada. Using the newly developed functional description as a baseline, the system was redesigned by two separate teams. One team concentrated on redesigning the STAMIS using a functional decomposition methodology involving structured analysis and design techniques. The second project team based their redesign on object-oriented design (OOD) techniques. Each development team was assigned a specific workstation to build the designs, but the CASE tool maintained all the models within a single library database networked between the two workstations.

Both design efforts were done over a two month period. After comparing the two designs to assess their respective advantages/disadvantages and discussing the lessons learned applying the methodologies, the OOD was selected to implement the STAMIS. The

OOD was chosen because it was simpler to understand, appeared to be easier to implement and promised reduced maintenance effort during the life-cycle.

The Teamwork CASE tool was used to generate Ada Structure Graphs (ASGs) from the OOD. These are graphical representations of Ada constructs that give a high level view of the system. All design specifications (including the most detailed interfaces) were derived from these graphs.

The hardest part of the OOD task was identifying the objects within the system. OOD is a paradigm shift from traditional system development. The programmers/developers who had a background in problem-solving techniques and structured design methodologies experienced an easier transition to the OOD way of viewing systems. The design document was so easily understood that it was used as a basis for final validation of the functional requirements by the proponent agency (PA). Enhancements were added to the functional requirements during the design phase based on discussions with the functional users. An interactive user interface, automatic data validation and error checking, and context-sensitive help information were among the new requirements added to the system. After viewing the new system design, the PA submitted an Engineering Change Proposal (ECP) to implement the new design as a production system and to downsize the system to run on existing PC's, replacing the old mainframe version.

Further work is needed to define rules or procedures to aid designers in more easily identifying the objects within a system during the initial steps in object-oriented design.

1-4.5. RE-IMPLEMENTATION

The design documents, including the ASGs, the functional description, and the Ada template code, were given to the programmers at SDC-A to implement. The system as it was designed consisted of 10 major components, each corresponding to an Ada package. The programming was accomplished by 2 analysts in 4 1/2 months with 7 man-months of effort. Both programmers had attended an 8-week Ada training course, but had no prior experience building systems using Ada. One of the programmers had maintained the original version of the system and had an extensive background in COBOL, but no exposure to OOD. The other programmer was knowledgeable in problem-solving and structured methodologies.

The coding was done using the Ada library procedures available with the Alsys Ada compiler and locally written Ada packages. They also took advantage of the reusable

components available through the Army's RAPID (Reusable Ada Products for Information Systems Development) library to handle some low-level I/O routines. Reusable, generic modules that were developed by SDC-A programmers have been submitted to the RAPID center as candidates for the reuse library. The package bodies furnished by the CASE tool described all the interfaces and data necessary for each component, allowing a great deal of programmer flexibility in coding the procedures. The programmers only had to implement the algorithms to perform the functions based on the predefined specifications. The final system consisted of 12,000 lines of Ada code.

1-4.6. TEST AND EVALUATION

A system test plan was developed to establish a framework for system validation and acceptance as an operational STAMIS. The test plan was defined to demonstrate the operational effectiveness against the performance and functional requirements as outlined in DoD-STD-7935A,"Automated Information Systems Documentation Standards".

SDC-A conducted an in-house Software Qualification Test (SQT) in May of 1991. An Army installation was chosen for initial live-site testing in May 1991. The results from this testing were used to correct the few problems that were found. The formal software acceptance test (SAT) was conducted in June 1991. There were no errors uncovered during the acceptance testing and the system was approved for release.

1-4.7. INSTALLATION

System installation documentation and support utilities were written by the programmers during the implementation phase. Because the system was to run on microcomputers, the entire system, including documentation and source code, was sent to the 40 sites through the mail. Installing the system was very easy to accomplish and required a minimal amount of support from the developers. As of August 1991, the new STAMIS became an operational system at over 40 active and reserve Army installations. During the three months the system has been operational, there have been no errors reported.

The system has met with positive reactions from the functional user community. The decrease in overall turnaround time at each installation has been significant. One installation has reported that what had taken them up to 2 1/2 weeks to do with the old system, now can be completed in 35 minutes. Other sites reported similar reductions in turnaround time and an overall improvement in their business practices.

1-5. CONCLUSION

The re-engineering methodology used in this project resulted in a well-designed system that has improved the operations of a particular area for a large number of Army installations. Preliminary data from this re-engineering effort show the cost of re-engineering to be about $125,000. This includes tools, workstations, training, and manpower required to produce the new system.

Initial studies have shown that, as a result of improving the business practices, it is estimated that each installation would save approximately $80,000 over an assumed 10 year life of the system. Multiplied by 40 installations, the cost savings would be over $3,000,000.

Additional savings in development center system maintenance are anticipated due to improvements in the system design structure, use of CASE tools, and improved reliability.

One of the questions that arises is whether the success of this effort is repeatable. We plan to re-engineer a larger STAMIS in order to assess the repeatability and scalability of these methods. We are also concerned with the evaluation of alternative designs. We chose the object-oriented design because it appeared to be the best. We are now applying software design metrics against the old and new systems to serve as a quantifiable basis of design comparison.

Finally, we are performing a study to develop cost/benefit analysis methods to use as a guide for determining which applications should be selected for future re-engineering activities based on predicted cost savings.

During our next re-engineering task, we will focus more effort on capturing metric data as we go through the process. For example, we will conduct function point analysis, define input/output primitives, track lines-of-code produced by man-hour, and apply other software productivity metrics and complexity metrics to enable us to better assess the cost-benefit of re-engineering existing applications.